CAS Front-End Engineering

OBJECT-ORIENTED DESIGN AND PROGRAMMING

INTRO

Silvan Gehrig

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

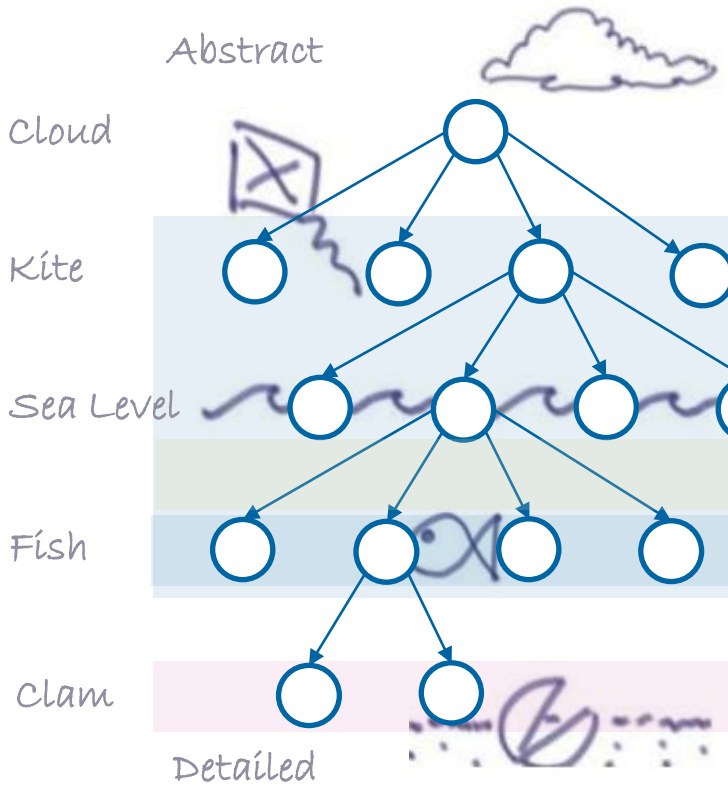
FHO Fachhochschule Ostschweiz

Table of Contents

- **Object-Oriented Analysis**
- **OOD Basics**
  - Einführung in die generellen Prinzipien
- **Übungen**
  - Zeit für Besprechung und Fragen

OBJECT-ORIENTED ANALYSIS

From Analysis to Design



Vision

Business Modeling

Requirement Modeling

Software Design

Programming

Object-Oriented Analysis

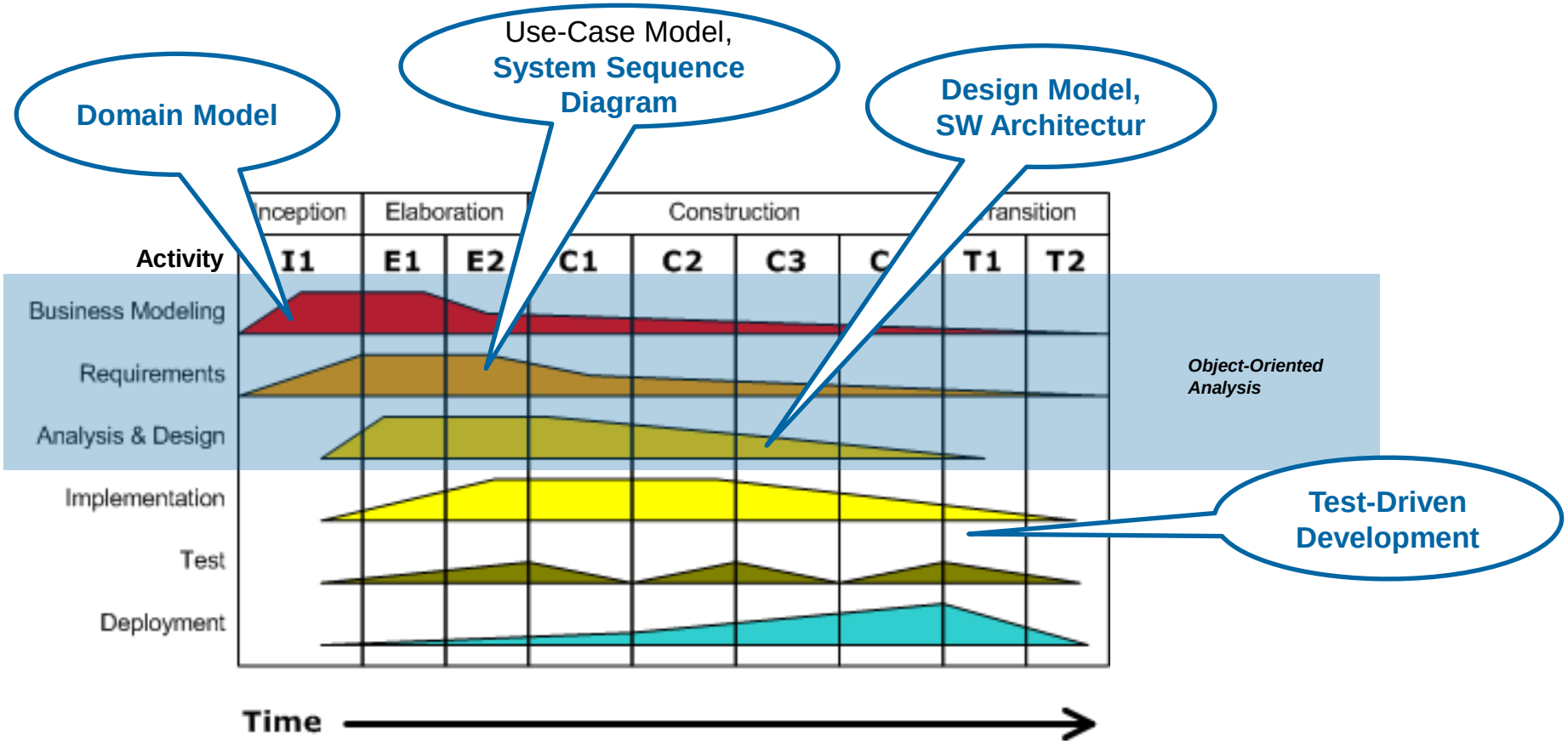
Object-Oriented Architecture

Object-Oriented Design

Object-Oriented Programming

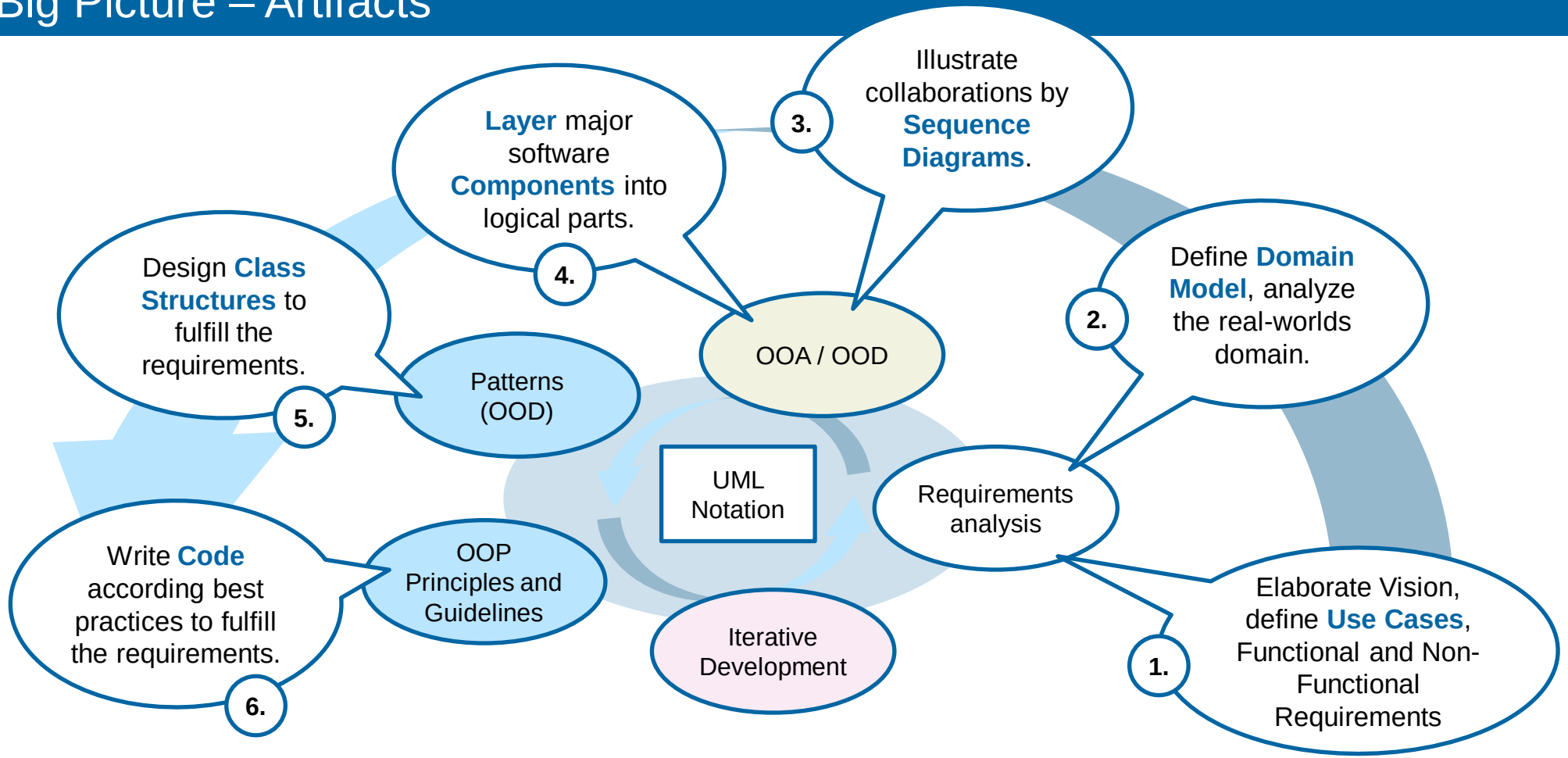
Source: Alistair Cockborn, Humans and Technology

UP Phases and Artifacts



Source: wikipedia.org

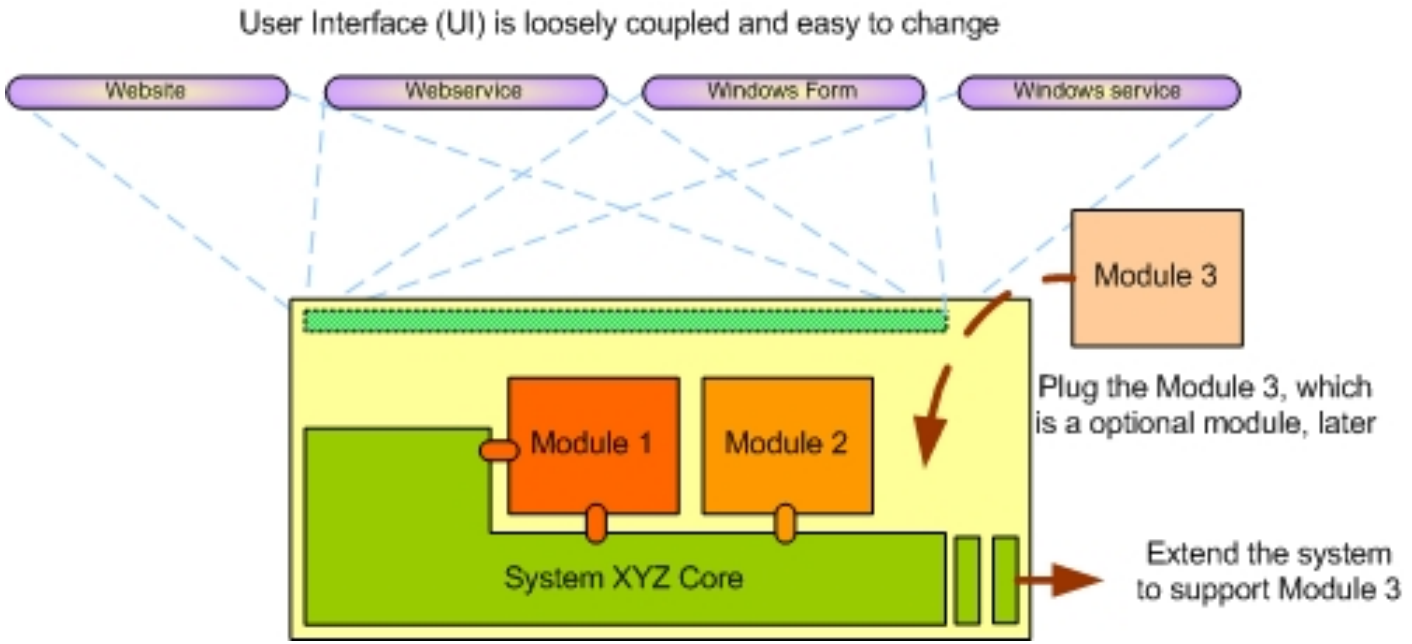
The Big Picture – Artifacts



OOD BASICS

What is Object-Oriented Architecture?

- **Partitioning the problem and the system**
- **Create interfaces between these pieces**
- **Manage overall structure and flow**



What is OOP – Object-Oriented Programming?

- **OOP is a design philosophy**
  - OOP is about how objects communicate
  - OOD (Object Oriented Design) is about how objects depend on each other
- **Everything in OOP is grouped into "objects".**
- **Defines software objects and how they collaborate to fulfill the requirements**

- **Leads to…**
  - Reusability of objects and responsibilities (Information Experts)
  - Higher cohesion between the objects
  - Lower coupling, reduce the impact of changes

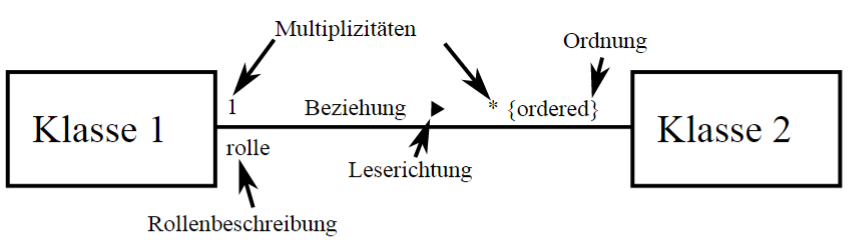
… more on OOP patterns by GRASP

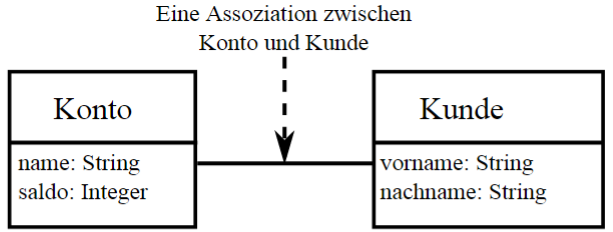
What is an «Object» / «Class»

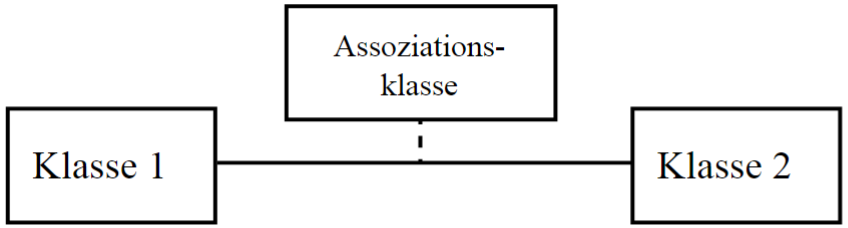
- **Object**
  - "thing" that can perform a set of related activities
  - Set of activities defines the object's behavior
  - An object instance is based on the type definition of a Class

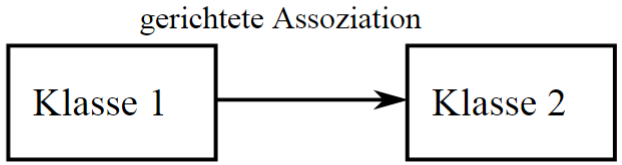
- **Class**
  - Representation of a type of objects
  - Blueprint, or plan, or template, that describes the details of an object
  - Class is composed of three things: a name, attributes, and operations

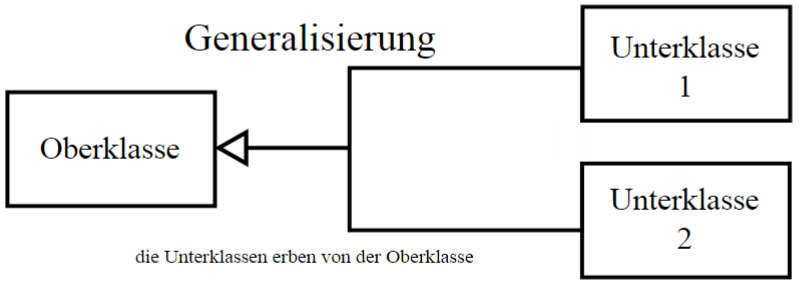
UML – a short introduction

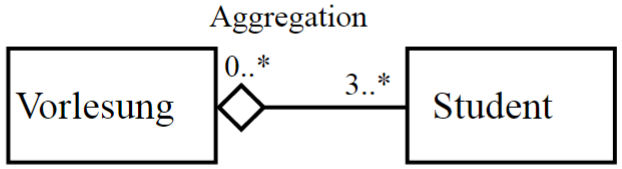


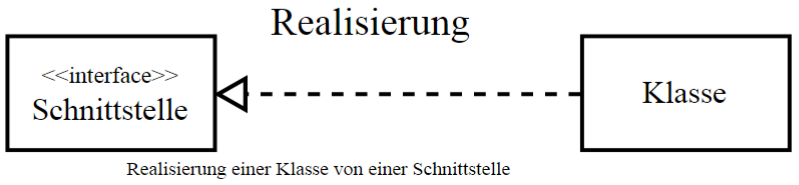


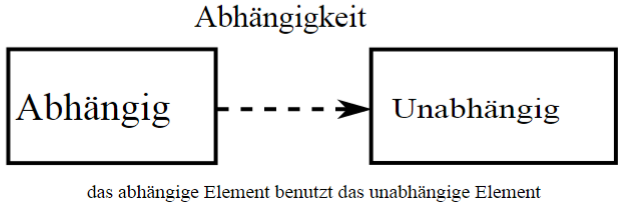












Auszug aus dem Pflichtenheft (Notenverwaltung): ***20 min***

Requirement	Description	
R023	Ein Student hat einen Namen und mehrere Noten.	MUST
R024	Noten können als mündliche Note und als Prüfung vorliegen.	MUST
R025	Die mündliche Note wird von zwei Experten, jeweils zu 1/3 und 2/3 festgelegt.	MUST
…		
R030	Eine Note ist einem Fach zugewiesen.	SHOULD

Erstellen Sie ein Klassendiagramm (Domain Model) mit den Klassen für die MUST Anforderungen oberhalb.

Optional: Ergänzen Sie Ihr Diagramm mit der SHOULD Anforderung.

How do you identify and design classes? I

- **General Responsibility Assignment Software Patterns (GRASP) provide a way to identify classes**

Information Expert	The class has exactly one responsibility and the necessary data.
Creator	Who creates?
Controller	First object that receives and coordinates a system operation.
Low Coupling	How to reduce impact of change?
High Cohesion	Keep objects focused, understandable, and manageable.
Polymorphism	Who is responsible when behavior varies by type?
Pure Fabrication	Who is responsible when you are desperate and you don't want to violate Low Coupling and High Cohesion?
Indirection	How to assign responsibilities to avoid direct coupling?
Protected Variations	Identify points of predicted variation (or instability); assign responsibilities to create a stable interface around them.

How do you identify and design classes? II

nice to know

- **There are five principles that you must follow when design a class (SOLID principles)**
  - **S**RP - The Single Responsibility Principle
    A class should have one, and only one, reason to change.
  - **O**CP - The Open Closed Principle
    Should be able to extend any classes' behaviors, without modifying the classes.
  - **L**SP - The Liskov Substitution Principle
    Derived classes must be substitutable for their base classes.
  - **I**SP - The Interface Segregation Principle
    Make fine grained interfaces that are client specific.
  - **D**IP - The Dependency Inversion Principle
    Depend on abstractions, not on concretions.

Übung zu OOD

Bilden Sie die Anforderungen aus der UML – DEMO / OOA in ECMA Script Code ab. ***30 min***

```
class MyClass extends MyBaseClass
{
    constructor(argument) {
        super();
        this.property = argument;
    }
 }
```

Erstellen Sie ein beliebiges HTML-File und referenzieren Sie Ihren JavaScript Code darin.

Optional: Ergänzen Sie Ihre Lösung mit dem SHOULD Requirement.

ÜBUNGSBESPRECHUNG

QUESTIONS?

Zusatzübung / Freiwillig

Auszug aus dem Pflichtenheft (Zoo Futterverwaltung):

Requirement	Description	
R002	Ein Tier hat einen Namen.	MUST
R003	Der Zoo unterhält Panda-Bären und Löwen.	MUST
R004	Panda's fressen Bambus, Löwen werden mit Fleisch gefüttert.	MUST
R005	Im Notfall könnten die Löwen auch mit den Pandas gefüttert werden.	SHOULD
R006	Nicht jedes Futter sättigt den Löwen im gleichen Masse: • 5kg Rindfleisch sättigt am besten (ca. 5 Stunden) • 10kg Hühnerfleisch weniger (ca. 1 Stunde)	SHOULD
R007	Der Panda frisst ca. 1kg Bambus in 1 Stunde.	SHOULD
R008	Die Tiere werden hungrig und müssen periodisch mit Futter versorgt werden.	SHOULD

Sources

- **Script (siehe Moodle)**
  - Nirosh L.W.C. http://www.codeproject.com/Articles/22769/Introduction-to-Object-Oriented-Programming-Concep
- **Craig Larman – Applying UML and Patterns**